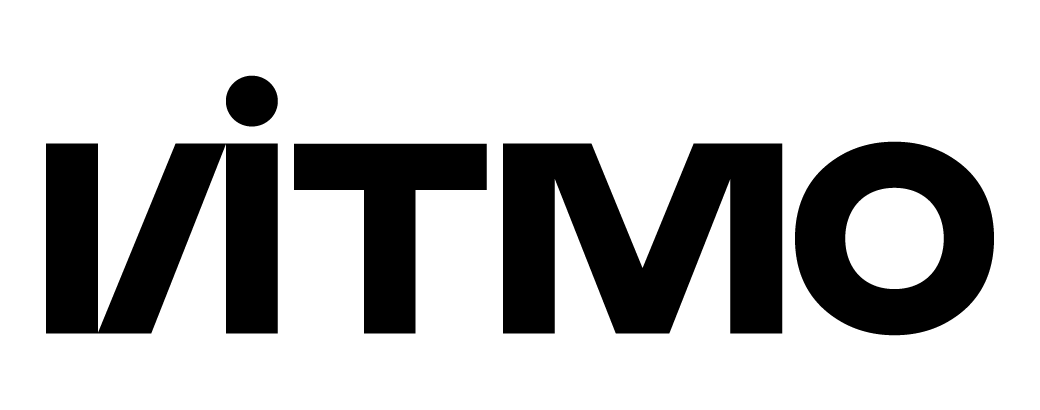
**инистр науки и высшего образования Российской̆ Федерации**

**Федеральное государственное автономное образовательное учреждение высшего образования  
«Национальный исследовательский университет ИТМО»**



Факультет информационных технологий и программирования

Лабораторная работа №4

***Написание Unit тестов. Github Actions.***

**Выполнил студент группы № M3113**

Рифад МД Абу Хуссаин

**Проверил:**

Жуйков Артём Сергееви

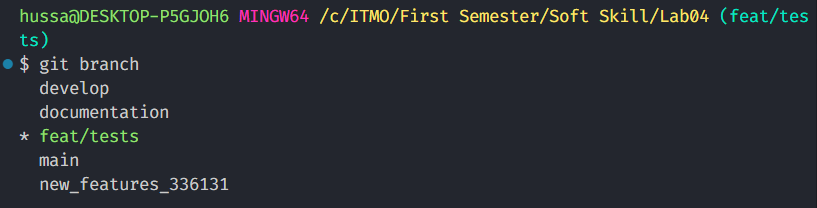
Санкт-Петербург

2024

**Part 01**

**1.**

Создаю новую ветку, в которой впоследствии и буду вести всю работу

****

**2.**

Функции для вычисления площади и периметра круга, квадрата и треугольника. Функция calc рассчитывает эти значения в зависимости от типа фигуры и аргументов, с проверкой ошибок.

import math

def area\_circle(r):

    """Вычисление площади круга"""

    return math.pi \* r \* r

def perimeter\_circle(r):

    """Вычисление периметра круга"""

    return 2 \* math.pi \* r

def area\_square(a):

    """Вычисление площади квадрата"""

    return a \* a

def perimeter\_square(a):

    """Вычисление периметра квадрата"""

    return 4 \* a

def area\_triangle(a, b, c):

    """Вычисление площади треугольника по формуле Герона"""

    s = (a + b + c) / 2

    return math.sqrt(s \* (s - a) \* (s - b) \* (s - c))

def perimeter\_triangle(a, b, c):

    """Вычисление периметра треугольника"""

    return a + b + c

def calc(shape, \*args):

    """Функция для вычисления площади или периметра заданной фигуры"""

    if shape == "circle":

        if len(args) != 1:

            raise ValueError("Круг должен иметь только один аргумент (радиус)")

        r = args[0]

        return {"area": area\_circle(r), "perimeter": perimeter\_circle(r)}

    elif shape == "square":

        if len(args) != 1:

            raise ValueError("Квадрат должен иметь только один аргумент (сторону)")

        a = args[0]

        return {"area": area\_square(a), "perimeter": perimeter\_square(a)}

    elif shape == "triangle":

        if len(args) != 3:

            raise ValueError("Треугольник должен иметь три аргумента (стороны)")

        a, b, c = args

        if a + b <= c or a + c <= b or b + c <= a:

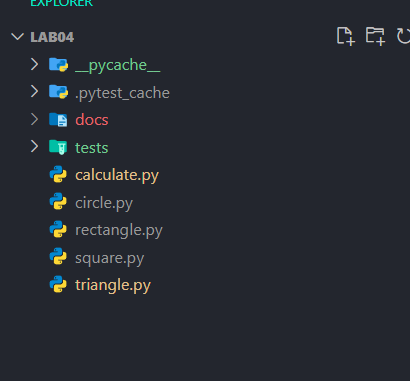
            raise ValueError("Сумма двух сторон треугольника должна быть больше третьей")

        return {"area": area\_triangle(a, b, c), "perimeter": perimeter\_triangle(a, b, c)}

    else:

        raise ValueError(f"Неизвестная фигура: {shape}")

**3.**

****

**4.**

Тесты с использованием pytest для вычисления площади и периметра круга, квадрата, треугольника и проверки функции calc, включая обработку ошибок для некорректных входных данных.

import pytest

import sys

import os

sys.path.insert(0, os.path.abspath(os.path.join(os.path.dirname(\_\_file\_\_), '..')))

from calculate import (

    area\_circle, perimeter\_circle,

    area\_square, perimeter\_square,

    area\_triangle, perimeter\_triangle,

    calc

)

# Тесты для круга

def test\_area\_circle():

    assert area\_circle(1) == pytest.approx(3.14159, 0.0001)

    assert area\_circle(0) == 0

    assert area\_circle(2) == pytest.approx(12.56637, 0.0001)

def test\_perimeter\_circle():

    assert perimeter\_circle(1) == pytest.approx(6.28318, 0.0001)

    assert perimeter\_circle(0) == 0

    assert perimeter\_circle(2) == pytest.approx(12.56637, 0.0001)

# Тесты для квадрата

def test\_area\_square():

    assert area\_square(2) == 4

    assert area\_square(0) == 0

    assert area\_square(5) == 25

def test\_perimeter\_square():

    assert perimeter\_square(2) == 8

    assert perimeter\_square(0) == 0

    assert perimeter\_square(5) == 20

# Тесты для треугольника

def test\_area\_triangle():

    assert area\_triangle(3, 4, 5) == pytest.approx(6, 0.0001)

    assert area\_triangle(6, 8, 10) == pytest.approx(24, 0.0001)

def test\_perimeter\_triangle():

    assert perimeter\_triangle(3, 4, 5) == 12

    assert perimeter\_triangle(6, 8, 10) == 24

# Тесты для функции calc

def test\_calc\_circle():

    result = calc("circle", 1)

    assert result["area"] == pytest.approx(3.14159, 0.0001)

    assert result["perimeter"] == pytest.approx(6.28318, 0.0001)

def test\_calc\_square():

    result = calc("square", 4)

    assert result["area"] == 16

    assert result["perimeter"] == 16

def test\_calc\_triangle():

    result = calc("triangle", 3, 4, 5)

    assert result["area"] == pytest.approx(6, 0.0001)

    assert result["perimeter"] == 12

def test\_calc\_invalid\_args():

    with pytest.raises(ValueError):

        calc("circle", 1, 2)

    with pytest.raises(ValueError):

        calc("triangle", 1, 2)

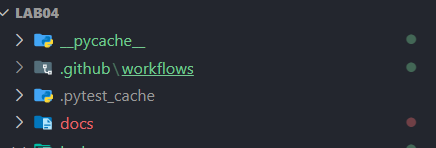
    with pytest.raises(ValueError):

        calc("unknown\_shape", 1)

**Part 02**

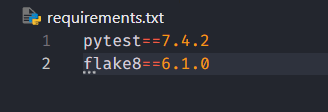
**1.**

Создайте файл воркфлоу



2.

Также нужно для определения зависимостей в проекте использую отдельный файл **requirements.txt**



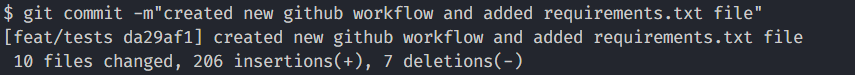
3.

Наконец создаю отдельную папку в проекте для различных рабочих пространств (workflows) и также создаю в ней файл с новым workflows для новых событий.



4.

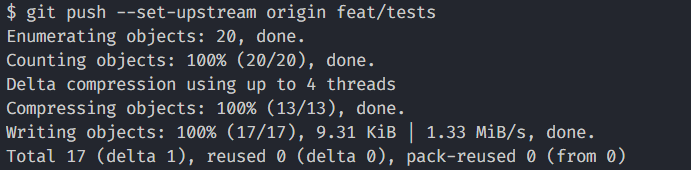
Коммит для проделанной работы



Part 03

1.

Push to github



2.

